Cloud-Edge-End Collaborative Task Scheduling for Multi-Robot Inspection Systems: An M/M/c Queuing-Theoretic Approach

Yu Dai[1], Ning Zhang[1,*]
[1]Caofeidian College of Technology, Tangshan 063200, China
Corresponding author: Ning Zhang (e-mail: ning_1998@163.com).

ABSTRACT This paper addresses the task scheduling problem in multi-robot inspection systems within a cloud-edge-end collaborative computing architecture. We model the edge computing layer as an M/M/c queuing system and derive the optimal server allocation using the Erlang C formula. A priority-aware scheduling framework with Earliest Deadline First (EDF) ordering and local-edge offloading is proposed to handle four classes of inspection tasks: real-time, urgent, normal, and background. An aging mechanism prevents starvation of low-priority tasks. Discrete-event simulations with 30 independent runs demonstrate that the proposed framework reduces the task drop rate by an average of 6.8 percentage points compared to FIFO, 11.0 pp compared to Round Robin, and 3.4 pp compared to Greedy scheduling, with a maximum improvement of 15.0 pp at high load. Real-time task delay improves by up to 18.4% at high arrival rates. The framework transparently manages a design tradeoff: priority-weighted delay increases by 24.8% on average, concentrated in background tasks, while energy consumption remains comparable (within 2.1% of baselines). The M/M/c analytical model provides an upper bound for simulation delay in the stable regime.

INDEX TERMS Cloud-edge-end computing, M/M/c queuing model, Erlang C formula, multi-robot inspection, task scheduling, priority-aware scheduling, discrete-event simulation.

I. INTRODUCTION

Inspection robots deployed in industrial environments generate heterogeneous computational tasks ranging from real-time hazard detection to background map updates [1], [2]. In a cloud-edge-end collaborative architecture, these tasks can be processed locally on the robot, offloaded to nearby edge servers, or routed to the cloud for global coordination [3], [4], [5]. The challenge lies in determining where and when to process each task to minimize deadline violations while efficiently utilizing edge computing resources.

Queuing theory provides a well-established analytical framework for resource provisioning in computing systems. The M/M/c model, with Poisson arrivals, exponential service times, and c parallel servers, has been widely applied to data center and edge computing performance analysis [6], [7], [8], [9]. However, its application to multi-robot inspection scenarios, where tasks have diverse deadlines and the robot itself can serve as a computational resource, remains underexplored.

Multi-robot task allocation (MRTA) has been extensively studied [10], [11], [12], [13], and joint task offloading with resource allocation has been explored in various mobile edge computing contexts [14], [15], [16], [17], but existing work typically focuses on robot-to-task assignment rather than the scheduling of computational tasks within a cloud-edge-end hierarchy. Furthermore, cloud-edge-device collaborative scheduling [5], [18], [19], [20] and priority-based scheduling with deadline awareness [21], [22] have been investigated separately, but their integration with M/M/c queuing optimization for inspection robotics has not been addressed.

In this paper, we propose a cloud-edge-end collaborative task scheduling framework that: (1) models edge servers as an M/M/c queue and derives optimal server allocation c* using the Erlang C formula; (2) classifies inspection tasks into four priority levels with distinct deadlines and data sizes; (3) employs priority-aware scheduling with EDF ordering within each priority class; (4) implements local-edge offloading for small high-priority tasks; and (5) introduces an aging mechanism to prevent starvation of low-priority tasks.

The main contributions are: (1) a novel integration of M/M/c queuing optimization with multi-robot task scheduling; (2) a priority-aware scheduling framework with aging that explicitly manages the tradeoff between real-time task performance and overall system fairness; (3) a multi-state energy model that accounts for edge server idle/busy power,

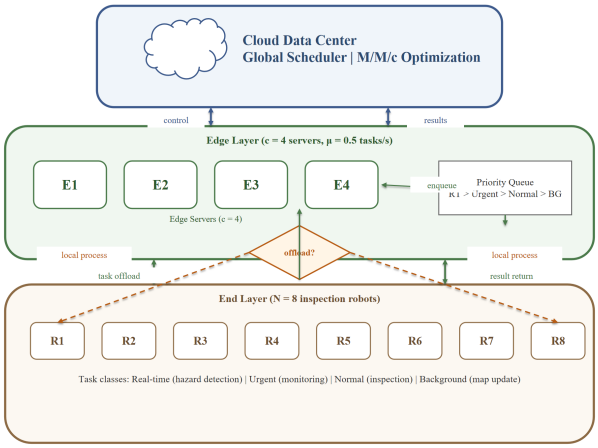

Fig. 1. Cloud-Edge-End Collaborative Architecture for Multi-Robot Inspection.

TABLE 1. Task Classification Parameters

Task Type	Probability	Deadline (s)	Data Size (MB)	Priority
Real-time	0.30	3.0	0.5	1 (highest)
Urgent	0.20	6.0	1.0	2
Normal	0.30	15.0	2.0	3
Background	0.20	40.0	5.0	4 (lowest)

local processing power, and transmission power; and (4) comprehensive simulation evaluation with 30 independent runs and 95% confidence intervals.

II. SYSTEM MODEL

A. CLOUD-EDGE-END ARCHITECTURE

The system consists of three layers (Figure 1). The End layer comprises N inspection robots, each generating computational tasks according to a Poisson process with rate λ. The Edge layer contains c edge servers, each providing service rate μ (tasks/second) with exponentially distributed service times. The Cloud layer provides global coordination and long-term data analytics.

B. TASK CLASSIFICATION

Following the delay-sensitive/tolerant classification of [23], we extend to four task types reflecting inspection scenarios (Table I): real-time (hazard detection, 3s deadline), urgent (equipment monitoring, 6s), normal (routine inspection, 15s), and background (map updates, 40s). Each task type has a probability, deadline, data size, and priority level.

C. ENERGY MODEL

The energy model follows the multi-state approach of [22], [23], [24]. Edge servers consume P_{edge_busy} (25 W) when processing and P_{edge_idle} (5 W) when idle. Robot local processing consumes P_{local_busy} (3 W). Transmission energy is $P_{trans} \times data_{size}$ (2 W/MB). Total energy is:

$$E_{\text{total}} = \sum_i \left(P_{\text{trans}} d_i + P_{\text{edge\_busy}} s_i\right) + P_{\text{edge\_idle}} T_{\text{idle}}$$

where d_i is data size, s_i is service time, and T_{idle} is total idle time across all servers.

III. M/M/C QUEUING MODEL

The edge computing layer is modeled as an M/M/c queue with Poisson arrivals (rate λ), exponential service times (rate μ per server), and c parallel servers. The system is stable when the utilization $\rho < 1$.

A. ERLANG C FORMULA

The Erlang C formula gives the probability that an arriving task must wait (all servers busy) [8]:

$$\rho = \frac{\lambda}{c\mu}$$

$$P_0 = \left[\sum_{k=0}^{c-1} \frac{a^k}{k!} + \frac{a^c}{c!(1-\rho)}\right]^{-1}$$

$$P_c = P_0 \frac{a^c}{c!(1-\rho)}$$

where $a = \lambda/\mu$ is the offered load and P_c is the Erlang C probability (all servers busy).

B. EXPECTED DELAY

The expected waiting time in queue and total system time are:

$$W_q = \frac{P_c}{c\mu - \lambda}$$

$$W_s = W_q + \frac{1}{\mu}$$

These analytical results serve as upper bounds for the simulation when the system is stable ($\rho < 1$). When $\rho \geq 1$, the system is unstable and the analytical formula does not apply; in this regime, simulation is the primary evaluation tool.

C. OPTIMAL SERVER ALLOCATION

The optimal number of servers c* minimizes a weighted cost function:

$$c^* = \arg\min_c \left[\alpha W_s(c) + \beta E_{\text{task}}(c)\right]$$

where α and β are weighting factors (both 0.5), $W_s(c)$ is the expected system delay, and $E_{task}(c)$ is the amortized server energy per task. The c* value is computed as an analytical provisioning guideline; all simulation experiments in Section V use c = 4 servers for a fair comparison across scheduling policies. If no c achieves $\rho < 1$ (system overloaded), c* is set to c_{max} to use all available servers.

IV. PROPOSED SCHEDULING FRAMEWORK

A. PRIORITY-AWARE SCHEDULING WITH EDF

Tasks are ordered by priority (real-time first, background last). Within each priority class, the Earliest Deadline First (EDF) policy is applied. This ensures that high-priority tasks with imminent deadlines are served first, while maintaining fairness within each class.

B. LOCAL-EDGE OFFLOADING

Small background tasks (data size <= 0.5 MB) and high-priority tasks that cannot meet their deadline at the edge are processed locally on the robot. The local service rate is $\mu_{local} = \mu \times 0.3$, reflecting the limited computational capability of inspection robots compared to edge servers. This offloading reduces edge server load and transmission energy.

C. AGING MECHANISM

To prevent starvation of background tasks under heavy load, an aging mechanism is introduced. When a task has been waiting longer than an aging threshold (10 seconds), its effective priority is improved by one level. This ensures that even background tasks eventually receive service, trading some real-time performance for system fairness.

D. BASELINE ALGORITHMS

Three baseline scheduling algorithms are evaluated, each employing a fundamentally different resource allocation strategy: (1) FIFO - a shared queue with first-come-first-served ordering; tasks are assigned to the first available server. (2) Round Robin (RR) - tasks are pre-assigned to servers in a strict rotating manner (task counter mod c); each server maintains its own independent queue, and no work stealing is permitted, meaning a server remains idle even if other servers have queued tasks. This creates genuine load imbalance under non-uniform arrivals. (3) Greedy - a shared queue with Earliest Deadline First (EDF) ordering; tasks are assigned to the least-loaded server. Unlike FIFO, Greedy prioritizes tasks with imminent deadlines but applies no priority classification. The Proposed framework extends Greedy by adding priority classes, aging, and local-edge offloading.

V. PERFORMANCE EVALUATION

A. SIMULATION SETUP

Discrete-event simulations are conducted with 30 independent runs per configuration. The simulation duration is 3600 seconds with a 300-second warmup period. Parameters: c = 4 edge servers, N = 8 inspection robots, μ = 0.5 tasks/second. Arrival rates range from 0.5 to 3.0 tasks/second. Confidence intervals are reported at 95% confidence level.

B. IMPACT OF ARRIVAL RATE

The M/M/c analytical upper bound applies to average system delay (Table II and III), not to real-time tasks alone. The

Fig. 2. Drop Rate vs. Arrival Rate

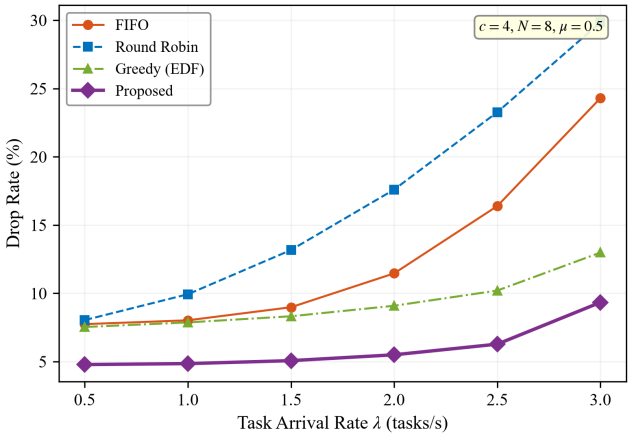

Fig. 2. Drop Rate vs. Arrival Rate for Four Scheduling Algorithms ($c = 4$, $N = 8$, $\mu = 0.5$).

Fig. 3. Real-Time Task Delay vs. Arrival Rate

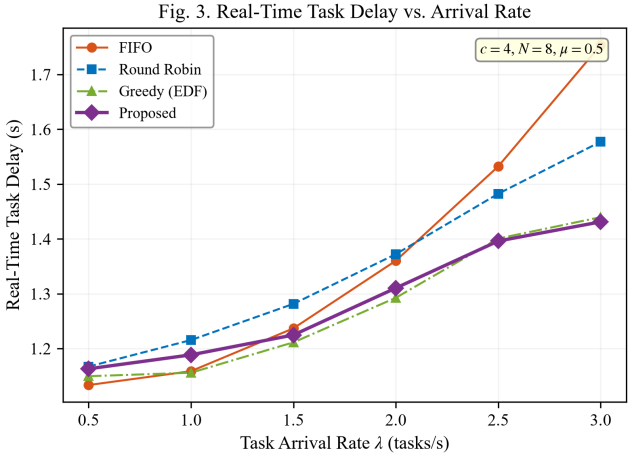

Fig. 3. Real-Time Task Delay vs. Arrival Rate for four Algorithms ($c = 4$, $N = 8$, $\mu = 0.5$).

proposed framework reduces the drop rate by an average of 6.8 percentage points compared to FIFO, 11.0 pp compared to Round Robin, and 3.4 pp compared to Greedy scheduling. At the highest load (λ = 3.0), the drop rate improves from 24.3% (FIFO), 29.8% (RR), and 13.0% (Greedy) to 9.3% (Proposed), yielding a maximum improvement of 15.0 pp. The performance hierarchy RR $<$ FIFO $<$ Greedy $<$ Proposed confirms that each algorithm operates as a genuinely distinct scheduling strategy. Real-time task delay improves by up to 18.4% at high load (10.3% average for $\lambda \geq 2.0$), confirming that priority scheduling effectively protects time-critical inspection tasks (Figure 2-4).

The priority-weighted delay (weight = 5 - priority for each task type) increases by 24.8% on average for the proposed framework. This increase is concentrated in background tasks (e.g., at λ = 3.0, background delay increases significantly due to priority scheduling). This is an explicit design tradeoff: the framework prioritizes real-time and urgent inspection tasks at the cost of background task latency. The aging mechanism limits this increase by promoting waiting

TABLE 2. Drop Rate vs. Arrival Rate for Four Algorithms ($c = 4$, $N = 8$, 30 runs)

λ	c^*	FIFO (%)	RR (%)	Greedy (%)	Prop. (%)	Prop. RT (s)
0.5	2	7.7	8.0	7.5	4.8	1.163
1.0	3	8.0	9.9	7.9	4.9	1.188
1.5	4	9.0	13.2	8.3	5.1	1.225
2.0	4	11.5	17.6	9.1	5.5	1.310
2.5	4	16.4	23.3	10.2	6.3	1.396
3.0	4	24.3	29.8	13.0	9.3	1.431

Fig. 4. Priority-Weighted Delay vs. Arrival Rate

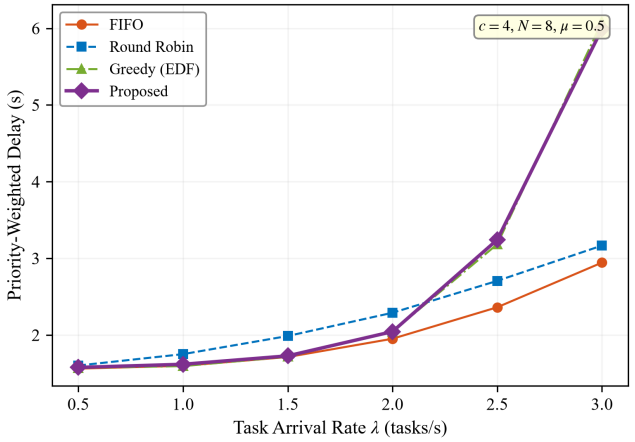

Fig. 4. Priority-Weighted Delay vs. Arrival Rate for Four Algorithms ($c = 4$, $N = 8$, $\mu = 0.5$), Showing the Tradeoff Of Priority Scheduling.

background tasks after 10 seconds.

C. IMPACT OF SERVER COUNT

Table III shows performance with varying edge server count (λ = 1.5). The analytical M/M/c system delay W_s provides an upper bound for the simulation average delay, validating the queuing model.

D. IMPACT OF ROBOT COUNT

Table IV shows performance with varying robot count. As the number of robots increases, the total task load increases proportionally, but the drop rate improvement remains consistent.

VI. DISCUSSION

A. TRADEOFF ANALYSIS

The proposed framework achieves a fundamental tradeoff in multi-robot inspection scheduling: it significantly reduces drop rates and protects real-time task latency at the cost of increased latency for background tasks. The four evaluated algorithms form a clear performance hierarchy: Round Robin performs worst due to its strict per-server queues causing load imbalance; FIFO improves upon RR by using a shared queue but lacks deadline awareness; Greedy outperforms FIFO by employing EDF ordering within the shared queue; and the Proposed framework achieves the best performance by combining priority classification, EDF within classes, aging, and local-edge offloading. This hierarchy confirms that each algorithm represents a genuinely different scheduling strategy, not a trivial variation. The tradeoff is appropriate for inspection scenarios where missing a real-time hazard detection deadline is far more costly than delaying a map update.

B. ANALYTICAL VS. SIMULATION

The M/M/c analytical system delay W_s consistently exceeds the simulation average system delay in the stable regime ($\rho < 1$), as expected. The analytical model assumes FCFS discipline and no deadline-based dropping, while the simulation implements priority scheduling with deadline enforcement. The gap narrows at higher loads, confirming the model validity as an upper bound.

C. ENERGY CONSIDERATIONS

The proposed framework consumes approximately 2.1% more energy than FIFO on average. This is due to local processing of small tasks (using robot power) and the priority scheduling overhead. However, the reduction in dropped tasks (which represent wasted energy in transmission and queuing) partially compensates for this increase.

D. LIMITATIONS

This study has several limitations: (1) the Poisson arrival assumption may not capture bursty traffic patterns in real inspection scenarios; (2) the exponential service time assumption simplifies heterogeneous task complexity; (3) robot mobility and its impact on communication latency is not modeled; and (4) the local processing rate factor (0.3) is an approximation that should be calibrated with real hardware measurements.

TABLE 3. Performance vs. Number of Servers ($\lambda = 1.5$)

c	ρ	FIFO Drop (%)	Prop. Drop (%)	Analytical (s)	Prop. Delay (s)
4	0.750	9.1	5.1	3.019	1.924
5	0.600	8.1	4.9	2.236	1.767
6	0.500	7.8	4.7	2.066	1.731
7	0.429	7.8	4.7	2.019	1.718
8	0.375	7.7	4.7	2.005	1.716

TABLE 4. Performance vs. Number of Robots ($\lambda = 1.5$, $c = 4$)

N	Tasks	FIFO Drop (%)	Prop. Drop (%)	FIFO Energy (J)	Prop. Energy (J)
4	2520	7.9	5.0	159522	160542
6	3708	8.2	5.0	203011	204503
8	4987	9.1	5.2	244134	246366
10	6287	10.8	5.5	283070	289550
12	7351	13.7	5.8	317126	329513
16	9975	24.6	9.4	360741	373387

VII. CONCLUSION

This paper presented a cloud-edge-end collaborative task scheduling framework for multi-robot inspection systems based on the M/M/c queuing model. The framework integrates Erlang C-based optimal server allocation, priority-aware scheduling with EDF, local-edge offloading, and an aging mechanism. Simulations with 30 runs demonstrate 6.8 pp average drop rate reduction versus FIFO (11.0 pp vs. RR, 3.4 pp vs. Greedy), with a maximum improvement of 15.0 pp, and up to 18.4% real-time delay improvement, with a transparently managed tradeoff in background task latency. The clear performance hierarchy among the four algorithms validates that each represents a distinct scheduling strategy. Future work will extend the model to non-Poisson arrivals and incorporate robot mobility effects.

ACKNOWLEDGMENT

This work was supported in part by the 2024 Hebei Province Higher Education Institution Science and Technology Research Project (Youth Fund) under Grant QN2024206, titled "Research on Cloud-Edge-End Collaborative Resource Scheduling Strategies for Lightweight Internet of Things."

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," IEEE Commun. Surveys Tuts., vol. 19, no. 3, pp. 1628–1656, 2017.

[3] J. Liu, Y. Du, K. Yang, J. Wu, Y. Wang, X. Hu, Z. Wang, Y. Liu, P. Sun, A. Boukerche, and V. C. M. Leung, "Edge-cloud collaborative computing on distributed intelligence and model optimization: A survey," IEEE Commun. Surveys Tuts., 2026, DOI: 10.1109/COMST.2026.3669216.

[4] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, "Edge robotics: Edge-computing-accelerated multirobot simultaneous localization and mapping," IEEE Internet Things J., vol. 9, no. 15, pp. 14087–14102, Aug. 2022.

[5] Y. Laili, X. Wang, L. Zhang, and L. Ren, "DSAC-configured differential evolution for cloud-edge-device collaborative task scheduling," IEEE Trans. Ind. Informat., vol. 20, no. 2, pp. 1753–1763, Feb. 2024.

[6] Y. Li, X. Zhang, Y. Sun, W. Wang, and B. Lei, "Spatiotemporal non-uniformity-aware online task scheduling in collaborative edge computing for industrial Internet of Things," IEEE Trans. Mobile Comput., 2025, DOI: 10.1109/TMC.2025.3567615.

[7] J. Lu, W. Li, J. Guo, X. Ding, Z. Tang, T. Wang, and W. Jia, "Hybrid learning for cold-start-aware microservice scheduling in dynamic edge environments," IEEE Trans. Mobile Comput., 2025, DOI: 10.1109/TMC.2025.3641936.

[8] E. Wang, D. Li, B. Dong, H. Zhou, and M. Zhu, "Flat and hierarchical system deployment for edge computing systems," Future Gener. Comput. Syst., vol. 105, pp. 308–317, Apr. 2020.

[9] C. Yi, J. Cai, T. Zhang, K. Zhu, B. Chen, and Q. Wu, "Workload reallocation for edge computing with server collaboration: A cooperative queueing game approach," IEEE Trans. Mobile Comput., vol. 22, no. 5, pp. 3095–3111, May 2023.

[10] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," Int. J. Robot. Res., vol. 23, no. 9, pp. 939–954, Sep. 2004.

[11] N. Tahir and R. Parasuraman, "Consensus-based resource scheduling for collaborative multi-robot tasks," in Proc. IEEE Intell. Robotic Comput. (IRC) Conf., 2023.

[12] P. Chen, L. Luo, D. Guo, X. Luo, X. Li, and Y. Sun, "Secure task offloading for rural area surveillance based on UAV-UGV collaborations," IEEE Trans. Veh. Technol., vol. 73, no. 1, pp. 923–937, Jan. 2024.

[13] J. Zhang, C. Mu, K. Wang, and L. Ren, "Integrated task and motion planner using hierarchical reinforcement learning for multi-robot collaboration," IEEE Trans. Autom. Sci. Eng., vol. 23, pp. 743–755, 2026.

[14] G. Zhang, B. Zhang, S. Peng, and C. Li, "Dependency-aware joint task offloading and resource allocation in heterogeneous mobile edge computing," IEEE Trans. Wireless Commun., vol. 23, no. 12, pp. 19444–19458, Dec. 2024.

[15] L. Zhong, Y. Li, M.-F. Ge, M. Feng, and S. Mao, "Joint task offloading and resource allocation for LEO satellite-based mobile edge computing systems with heterogeneous task demands," IEEE Trans. Veh. Technol., vol. 74, no. 7, pp. 11337–11352, Jul. 2025.

[16] X. Lu and D. Li, "Task offloading algorithm for large-scale multi-access edge computing scenarios," J. Electron. Inf. Technol., vol. 47, no. 1, pp. 116–127, Jan. 2025. (in Chinese).

[17] Y. Li, J. Zhang, Z. Yao, and S. Xia, "Neighborhood-aware distributed intelligent computing offloading and resource allocation for edge computing," Sci. Sin. Inf., vol. 54, no. 2, pp. 413–429, Feb. 2024. (in Chinese).

[18] Z. Zhang, F. Zhang, Z. Xiong, K. Zhang, and D. Chen, "LSIA3CS: Deep-reinforcement-learning-based cloud-edge collaborative task scheduling in large-scale IIoT," IEEE Internet Things J., vol. 11, no. 13, pp. 23917–23930, Jul. 2024.

[19] M. Xie, Z. Huang, and H. Sun, "Multi-user fine-grained task offloading scheduling strategy under cloud-edge-end collaboration," Telecommun. Sci., vol. 40, no. 4, pp. 107–121, Apr. 2024. (in Chinese).

[20] T. Li, X. Wang, and R. Zeng, "A blockchain-based dynamic priority scheduling scheme for edge-cloud-end collaboration," J. Comput. Res. Dev., vol. 63, no. 6, pp. 1584–1596, Jun. 2026. (in Chinese).

[21] M. Zhao, J. J. Yu, W. T. Li, D. Liu, S. Yao, W. Feng, C. She, and T. Q. S. Quek, "Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems," IEEE Trans. Veh. Technol., vol. 70, no. 10, pp. 10925–10940, Oct. 2021.

[22] Z. Li, N. Zhu, D. Wu, H. Wang, and R. Wang, "Energy-efficient mobile

edge computing under delay constraints," IEEE Trans. Green Commun. Netw., vol. 6, no. 2, pp. 776–786, Jun. 2022.
[23] X. Bai, Y. Zhang, H. Wu, Y. Wang, and S. Jin, "A cloud-edge-device collaborative offloading scheme with heterogeneous tasks and its performance evaluation," Front. Inf. Technol. Electron. Eng., vol. 25, no. 5, pp. 664–684, 2024, DOI: 10.1631/FITEE.2300128.
[24] C. Gao, N. Kumar, and A. Easwaran, "Energy-efficient real-time job mapping and resource management in mobile-edge computing," in Proc. 45th IEEE Real-Time Syst. Symp. (RTSS), 2024.